(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 October 2001 (18.10.2001)

PCT

(10) International Publication Number
WO 01/78420 A1

(51) International Patent Classification⁷: H04Q 3/64, H04L 12/56

(21) International Application Number: PCT/SE01/00733

(22) International Filing Date: 4 April 2001 (04.04.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/546,494 10 April 2000 (10.04.2000) US

(71) Applicant *(for all designated States except US)*: SWITCHCORE AB [SE/SE]; Scheelevägen 32, S-223 63 Lund (SE).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: AHLFORS, Ulf [SE/SE]; Sunnanvag 19 C, S-244 38 Kävlinge (SE). FYHN, Anders [SE/SE]; Floragatan 8 A, S-212 21 Malmö (SE). TUFVESSON, Peter [SE/SE]; Kobjersvagen 9 A, S-227 38 Lund (SE).

(74) Agents: ÅKERMAN, Mårten et al.; c/o Albihns Malmö AB, P.O. Box 4289, S-203 14 Malmö (SE).

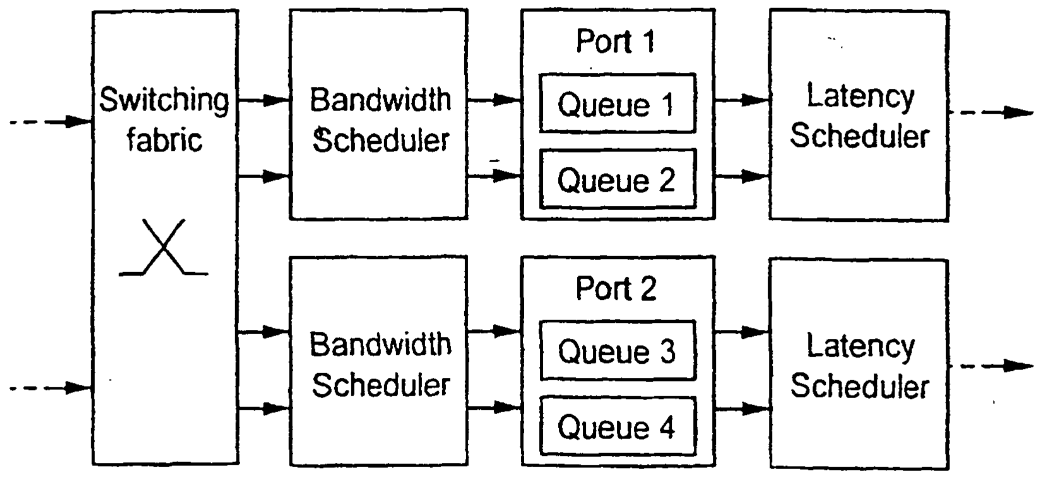
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR DISTRIBUTION OF BANDWIDTH IN A SWITCH



(57) **Abstract:** The invention relates to a method and an apparatus for distribution of bandwidth in a switch or router. More particularly, the invention relates to a scheduler and an associated algorithm for distributing bandwidth over data traffic directed to output ports and received in various traffic classes and flows. The switch comprises a switching fabric. Preferably, the bandwidth scheduler is located before output queues, and the method comprises: receiving a stream of data from the switching fabric; subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is forwarded or interrupted (accepted or rejected). Preferably, the stream of data includes identifiable data packets and the decision making algorithm in the bandwidth scheduler results in that the data packet is accepted or rejected. The bandwidth scheduler may be located before the output queues leading to early discarding of packets and efficient use of output buffer memory. The algorithm includes logical rules operating on counters and variables recording the accepted traffic to implement the bandwidth distribution. The algorithm enables weighted distribution and short term as well as long term fairness.

WO 01/78420 A1

METHOD AND APPARATUS FOR DISTRIBUTION OF BANDWIDTH IN A SWITCH

Field of the invention

The present invention relates to a method and an apparatus for distribution of bandwidth in a switch or router. More particularly, the invention relates to a scheduler and an associated algorithm for distributing bandwidth over data traffic directed to output ports and received in various traffic classes and flows. The bandwidth scheduler may be located before the output queues leading to early discarding of packets and efficient use of output buffer memory. The algorithm includes logical rules operating on counters and variables recording the accepted traffic to implement the bandwidth distribution. The algorithm enables weighted distribution and short term as well as long term fairness.

State of the art

The paper András Rácz, Gábor Fodor, Zoltán Turányi: *Weighted Fair Early Packet Discard at an ATM Switch Output Port*, 0-7803-55420-6/99 1999 IEEE discloses similar ideas on fairness and bandwidth utilisation in an ATM switch. A predefined weighted share is associated with a data stream. An algorithm attempts to provide this share of the bandwidth for the streams in the long time average. However, the paper is silent on the division of the scheduler in bandwidth and latency scheduling and also on many other aspects of the present invention.

One object of the present invention is to split the scheduler is into two parts, a bandwidth scheduler and a latency scheduler. Bandwidth scheduling is performed before packets arrive in the output queues. Packets eligible for dropping are pro-actively blocked. Thus, it is no longer necessary to differentiate flows and/or traffic flows in order to allocate bandwidth and the output queues can be used solely for latency priorities.

Summary of the invention

These and other objects of the invention are achieved by the present invention which provides a method and an apparatus for bandwidth scheduling in a switch comprising a switching fabric. In accordance with a first embodiment the bandwidth scheduler is located before output queues, and the method comprises: receiving a stream of data from the switching fabric; subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is forwarded or interrupted (accepted or rejected). Preferably, the stream of data includes identifiable data packets and the decision making algorithm in the bandwidth scheduler results in that the data packet is accepted or rejected.

In accordance with further embodiments, a number of logic rules and operations are run through including:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

the bandwidth is distributed in accordance with the Max-Min algorithm;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

In accordance with another embodiment the bandwidth scheduler is located after output queues.

One advantage of the present invention is that bandwidth is distributed much earlier, resulting in smaller buffer requirements and smaller buffer usage fluctuations. Also, the algorithm is totally independent of the number of output queues per port, while algorithms like Weighted Round Robin and Weighted Fair Queuing need as many queues as possible.

<u>Brief description of the drawings</u>

The invention will be described below with reference to the accompanying drawings, in which:

fig 1 is a block diagram of a prior art scheduler,

fig 2 is a block diagram of a split scheduler architecture according to the present invention,

fig 3 is a diagram of bandwidth distribution in accordance with the prior art Max-Min algorithm,

fig 4 is a diagram of accepted bandwidth using the backlogging and charity counters according to the present invention,

fig 5 is a diagram of the backlogging counter associated with figure 4,

fig 6 is a diagram of experienced bandwidth associated with figure 4, and

fig 7 is a diagram of a hierarchical structure of traffic at different levels.

Detailed description of preferred embodiments

Generally, the task of a scheduler is to forward or discard traffic received from a switching fabric to output ports and respective output links. The concept of Quality of Service has been introduced to define the quality of the operation of the switch. Four different aspects of Quality of Service may be studied. First is latency, the delay the flow is experiencing through the device. Second there is jitter, or latency variations. Third there is bandwidth distribution and fourth is loss probability. The present invention is mainly related to bandwidth distribution.

In figure 1, the prior art architecture with a combined latency and bandwidth scheduler is shown. Traffic is switched by a switching fabric and distributed on ports which may have a number of queues each. The scheduler is located after the output queues. Examples of this kind of scheduler are Round Robin, Weighted Round Robin and Weighted Fair Queuing. Here the queues are used to separate different flows and/or traffic classes so that the scheduler can differentiate them. This type of architecture uses common techniques like tail-drop or push-out to drop packets.

In figure 2 the scheduler architecture according to the present invention is shown. The main difference is that the scheduler is split into two parts, a bandwidth scheduler and a latency scheduler. Bandwidth scheduling is performed before packets arrive in the output queues. Packets eligible for dropping are pro-actively blocked. Thus, it is no longer necessary to differentiate flows and/or traffic flows in order to allocate bandwidth and the output queues can be used solely for latency priorities. One advantage is that bandwidth is distributed much earlier, resulting in smaller buffer requirements and smaller buffer usage fluctuations. Also, the algorithm is totally independent of the number of output queues per port, while algorithms like Weighted Round Robin and Weighted Fair Queuing need as many queues as possible.

Any latency scheduler can work together with the bandwidth scheduler according to the present invention and strict priority is proposed.

Another aspect of the present invention is the bandwidth scheduler algorithm as such. The algorithm aims at a fair distribution of the bandwidth between traffic classes and flows at each port. The algorithm takes into account many factors, such as the bandwidth demand of each flow, and short term and long term fairness as will be described more in detail below. The algorithm as such is general and may in principle be located before or after the output ports.

A fair bandwidth distribution can be accomplished in many different ways. Also fairness has different definitions and could be measured in various ways. Fairness could be defined as distributing a bandwidth equal to the wanted bandwidth divided by the sum of the wanted bandwidth. This can be accomplished by several

Round Robin schemes. However, in the present invention the Max-Min algorithm is preferred. As the name indicates, this algorithm maximizes the minimum flow. This is considered the fairest algorithm, if all flows can benefit equally to increased bandwidths.

The Max-Min algorithm is illustrated in figure 3. If the basic concept is that equal bandwidth is equal utility, then it is most fair to find a limit *l* were all flows that are offering less than *l* experience no losses. Flows that are offering more traffic only get bandwidth equal to *l*, no matter how much bandwidth they are offering. As seen from the figure, a fair share is defined for all flows. Since the fair share is not used by all flows, a spare bandwidth remains after fair share allocation. This spare bandwidth is distributed on flows offering more traffic than the fair share, up to the limit *l*. Flows offering more traffic than the limit *l* have this part of the traffic blocked.

The present invention proposes a further extension of the Max-Min algorithm:

First, all flows are not equal. Each flow is associated with a weight such that the bandwidth is distributed in relation to the weight of each flow. Preferably, each traffic class has a weight and the flows within a traffic class are treated equally.

Second, some flows can be guaranteed bandwidth. In other words, no data packets are lost until the flow exceeds the guaranteed bandwidth limit.

Third, some flows can be restricted to certain bandwidth maximum. Under no circumstances should a maximized flow get more bandwidth than its limit, even if the line will be left under-utilized.

Fourth, a short term fairness is introduced between flows. If a flow is bursty, i.e. more packets are sent than the accepted bandwidth, this should be accepted for a short period of time to make the scheduling flexible. The other flows will be compensated in the future.

Fifth, a long term fairness between flows is also introduced. If a flow is aggressive for a period it will be forced to give up some of its accepted bandwidth to the other flows as "charity". If a flow is silent for a time period, it will be compensated in the future by means of the accumulated charity, so that the flow is allocated more bandwidth than the competing flows. However, the time period should be limited and also the accumulated amount of compensation should be limited.

The implementation of the algorithm is described more in detail below.

The bandwidth scheduler generally receives a stream of data. The stream may be organized into cells or data packets according to different protocols, such as TCP (Transport Control Protocol) and UDP (User Datagram Protocol). The term data packet and similar in this application is intended to encompass any kind of data entity. It is also practical to use the term flow which can have different meanings under different circumstances. If e.g. TCP/IP is used the flow may be an application flow (address and port on both source and destination) or a host flow (only address of source and destination). It is assumed that each flow may be classified with regard to its identity with respect to the following categories.

The traffic is distributed on the respective ports. This is straightforward but usually the operator puts a limit on the maximum accepted bandwidth per port.

Each port may accommodate a number of traffic classes. All flows are categorised into classes. A class is normally based upon some network protocols and/or network hosts, but as regards the present invention the classes can be based upon any criteria. The classes must be fully disjoint and the invention does not have to be enabled for all classes. All flows within a traffic class are equal. If this is undesirable, a traffic class needs to be split up into two or more classes.

In principle, an application flow is the smallest unit treated by the scheduler. However, since the number of application flows is very large and seems to be growing at a rapid rate, the invention proposes to group application flows together by means of a hash function into a set of hashed groups which in this application by definition will be referred to as flow groups. The hashing function is stationary and deterministic in a way that all packets belonging to one flow always must be mapped into the same flow group. If flow groups are used, the invention does not distinguish between the flows within the flow group.

The physical implementation of the invention resides in a program stored in the scheduler either before or after the output queues. The program contains the algorithm defining logic rules operating on constants, configuration parameters and various variables and counters. The incoming data stream is stored in a buffer while the algorithm operates on some part of the data stream, for instance headers of individual data packets. The extracted information or header is processed through the algorithm and the result is that the data stream is forwarded or interrupted or, in case of a data packet, the packet is accepted or rejected. Various counters keep track of the accepted traffic for each traffic class and flow group. Also, the variables and counters are updated at regular intervals. The process is described in further details below, with reference to the various parts of the algorithm.

A number of parameters and variables are used to implement the alogorithm. They are listed in the tables below showing the hierarchical order of the variables and the rules for increasing, decreasing as well as updating the variables.

Configuration parameters

Port	Traffic class	Flow group
BWP_{max}	$BWTC_{max}$	
	$BWTC_{min}$	
	WTC	

BWP_{max}	maximum bandwidth per port
$BWTC_{max}$	maximum bandwidth per traffic class
$BWTC_{min}$	minimum bandwidth per traffic class
WTC	weight per traffic class

Port counters and variables

Name	Logic	Increment per packet sent/discarded	Update per time unit
VQLP		+packet length	$-BWP_{max}$
TCP_{max}	=max(all TCs of port)	—	—
BLP_{max}	=max(all BLs of port)	—	—
CH		+packet length × give factor, if given -packet length × WTC, if taken	× decay factor (e.g. 15/16)

VQLP	virtual queue length per port
TCP_{max}	maximum traffic class variable per port
BLP_{max}	maximum backlogging variable per port
CH	charity counter per port

Traffic Class counters and variables

Name	Logic	Increment per packet sent	Update per time unit
TC		+packet length × WTC	$-BWTC_{min}$ × WTC
FG_{max}	=max(all FGs of TC)	—	—
BL	0<BL<256 kB	+packet length × WTC	$-BWTC_{min}$ × WTC
VQLTC		+packet length	$-BWTC_{max}$

TC	traffic class counter
FG_{max}	maximum flow group variable per traffic class
BL	backlogging counter per traffic class
VQLTC	virtual queue length per traffic class

Flow Group counter

Name	Logic	Increment per packet sent	Update per time unit
FG		+packet length	—

FG	flow group counter

To illustrate the invention it is assumed that the data stream arrives in packets carrying information about flow identity. Each port receives its respective part of the data stream. The scheduler is configured to limit the amount of accepted bandwidth per port by means of a configuration parameter BWP_{max} (maximum bandwidth per port). To keep track of the accepted bandwidth for each port a virtual queue is implemented. In other words, a counter VQLP (virtual queue length of the port) is increased with the packet length when the port accepts a packet. By updating or refreshing the counter VQLP is each time unit by subtracting the configuration parameter BWP_{max}, the limit is maintained automatically. If the virtual queue grows too long (VQLP > constant), packets will be rejected.

As mentioned above, each port also usually accept traffic in various traffic classes. Each traffic class has a virtual queue length counter TC to keep track of the accepted bandwidth in each traffic class. A variable TCP_{max} is set at a value equal to the maximum of the traffic class counters for the port in question, to keep a record of the traffic class counter having the highest value. The counter TC is increased with the packet length when the traffic class accepts a packet. Also, the counter TC is updated or refreshed each time unit by subtracting a configuration parameter $BWTC_{min}$ (see below). The fairness may be defined in various ways. In the present invention the fairness is computed either as a difference or a ratio. Thus, a traffic class with the difference $TCP_{max} - TC <$ a constant, e.g. 128kB, is considered fair, while more busy classes are considered unfair. Alternatively, a traffic class with the ratio $TC/TCP_{max} <$ a constant, e.g. 0.75, is considered fair, while more busy classes are considered unfair. If the traffic class is fair, an offered packet may be accepted. If the virtual queue grows too long (TC > constant), unfair packets will be rejected. For the most aggressive traffic class ($TC = TCP_{max}$) offered packets are rejected when the virtual queue is even shorter. In this way the counter TC assists in implementing the basic algorithm Max-Min for the traffic classes.

Also each flow group has a virtual queue counter FG keeping track of how many packets are accepted. Each traffic class has a variable FG_{max} which is set equal to the maximum value of the counters FG belonging to this traffic class. A flow group with the difference $FG_{max} - FG <$ a constant, e.g. 128kB, is considered fair, while more busy flow groups are considered unfair. Alternatively, a flow group with the ratio FG/FG_{max} < a constant, e.g. 0.75, is considered fair, while more busy flow groups are considered unfair. For the most aggressive flow group ($FG = FG_{max}$) offered packets are rejected when the virtual queue is even shorter. In this way the counter FG assists in implementing the basic algorithm Max-Min for the flow groups.

The present invention involves a further extension of the Max-Min algorithm with the additions mentioned above. The additions operate in parallel and independently of one another. Not all the additions have to be implemented but may be combined in various ways.

To enable prioritizing of certain traffic classes over other, weights are associated with each traffic class. A configuration parameter WTC (weight traffic class) is set when initializing the scheduler. When packets are accepted the respective counters are increased in a weighted manner, so that the algorithm automatically prioritizes certain traffic classes. Thus, the counter TC is increased with the packet length multiplied by the weight WTC when the traffic class accepts a packet. Of course, the weight function may be disabled by setting all weights WTC to unity (1).

Each traffic class may be associated with a guaranteed bandwidth. A configuration parameter $BWTC_{min}$ (bandwidth traffic class minimum) is set when initializing the scheduler. If the traffic class in question offers bandwidth less than the guaranteed bandwidth, it will always be accepted. Of course, the total of the guaranteed bandwidth for all traffic classes must be less than or equal to the maximum bandwidth of the port BWP_{max}.

The counter TC is updated or refreshed each time unit by subtracting the configuration parameter $BWTC_{min}$ multiplied by the weight WTC. This is to account both for the weight and guaranteed bandwidth. This subtraction results in that all traffic below $BWTC_{min}$ for this class will be accepted. If the counter TC grows larger than $BWTC_{min}$ the traffic will compete equally with the other flows.

A maximum bandwidth may be associated with each traffic class. A configuration parameter $BWTC_{max}$ (bandwidth traffic class maximum) is set when initializing the scheduler. This parameter limits the amount of accepted traffic in a traffic class, irrespective of existing spare capacity. Another virtual queue is associated with each traffic class by means of a counter VQLTC (virtual queue length per traffic class) counting the number of accepted packets. The counter VQLTC is updated or refreshed each time unit by subtracting the configuration parameter $BWTC_{max}$. Thus, the limit is maintained automatically. If the virtual queue grows too long (VQLTC > constant possibly plus a tolerance constant to allow for different packet sizes), packets will be rejected.

To accommodate bursty traffic but still distribute bandwidth in a fair way seen over a short term a counter is introduced for each traffic class to keep a record of the amount of accepted traffic for one traffic class in relation to the other traffic classes belonging to the same port. The counters are called backlogging counters BL. Also, one variable BLP_{max} (backlogging port max) stores the maximum of the back-logging counters for the traffic classes of each port. A traffic class with the difference BLP_{max} - BL < a constant, e.g. 128kB, is considered fair, while more busy traffic classes are considered unfair. Alternatively, a traffic class with the ratio BL/BLP_{max} < a constant, e.g. 0.75, is considered fair, while more busy classes are considered unfair. The counter BL is increased with the packet length multiplied by the weight WTC when the traffic class accepts a packet. The counter BL is updated or refreshed each time unit by subtracting the configuration parameter $BWTC_{min}$ multiplied by the weight WTC. In this way the counter BL assists in implementing the basic algorithm Max-Min together with the counters TC and FG. This counter BL is associated with the concept of short term fairness, but the backlogging counter BL is also important for the weight function.

If a traffic class is idle for some time, the spare bandwidth is distributed among the active flows. When the idle flow becomes active again the flow is compensated by distributing more bandwidth to this flow. On the other hand, the now active class should not be allowed to monopolize the link in order to accomplish this. Instead this should be a slow process, given the quiet class a fraction more bandwidth until the flows are once again treated equally. On the other hand, if one traffic class is particularly aggressive or active, it should give up a part of its accepted bandwidth as "charity". Both these situations are associated with the concept of long term fairness. This feature is associated with a counter CH (charity) for each port (and added level, if any.) All levels under a port uses one and the same counter, and, in turn, all traffic classes under a level (port, if no added levels) uses one counter. When a packet is accepted in a traffic class having the maximum accepted bandwidth, in other words, the variable TC equals TCP_{max}, the packet may instead be discarded, if it is not unfair with regard to other criteria (depending on the queue length). Then, the counter CH is increased with a configurable fraction of the accepted packet length (+packet length × give factor). The other traffic class counters (TC and BL) are incremented as if the packet was accepted. On the other hand, when a packet is sent by one of the other traffic classes of which the counter TC ≠ TCP_{max}, and when the packet is decided to be rejected in accordance with the other logic rules, the traffic class can use the charity function to force the packet to be accepted. Then, the charity counter CH of a port is decreased with the packet length multiplied with the weight of the respective traffic class (or underlying level): -packet length × WTC (-packet length × WAL). Similarly, for an added level the charity counter CHAL is decreased with the packet length multiplied with the weight of the respective traffic class: -packet length × WTC. Thus, value of the charity counter CH will vary and reflects if one traffic class (or underlying level) is much more aggressive than the others. If the traffic classes are more or less equal, then the charity counter should preferably decay slowly. Thus, the counter CH is updated or refreshed each time unit by multiplying with a decay factor, e.g. 15/16.

Figure 4 is a first diagram showing the total accepted bandwidth and figure 5 is a diagram showing the backlogging counters for two flows A and B. The backlogging counter is increased every time a packet is accepted, such as shown in figure 4. The backlogging counter is limited to a fixed value, e.g. ±128 kB. If one backlogging counter for a flow reaches the upper limit, all the counters belonging to this port are decreased in order to maintain the internal difference. If one backlogging counter for a flow reaches the lower limit, this counter remains at the lower limit (but the other counters are not affected). This way only the near past is remembered. Hence the term short term fairness. Now we have two variables (backlogging counter and charity counter) measuring fairness in two different time scales.

Up to T1 two flows, A and B, are active. They are considered equal in all respects and offer the same amount of bandwidth to the switch. Between T1 and T2 only flow A is active, while flow B is idle. After T2 both flows are again active.

The two diagrams of figures 4 and 5 share the same time axis. Until T1 both flows have the same bandwidth and backlogging counters. Since flow B becomes idle at T1, only the counters of flow A are increased up to T2. Note that the backlogging counter has an upper limit and instead of continuing to increase, all flows are decreasing their backlogging counters. Also note that the backlogging counter has a lower limit, Min Backlogging in figure 5. Also the charity counter CH of the port is increased, since flow A is the most aggressive flow and discards some packets. When both flows A and B offer bandwidth, only the flow having the smallest backlogging counter BL is accepted. At T2 flow B becomes active again and for a small period, T2 to T3, all the traffic of flow B is accepted while the backlogging counters of flow B is increased. Once the backlogging counters are equal, they share the bandwidth again.

Between T3 and T4 the accepted bandwidth differs between flow A and B. Until they match, flow A is giving up a small portion of its bandwidth for flow B. Now the charity counter CH of the port is increased by flow A discarding some packets and decreased by flow B taking some packets. After T4 they share the line equally again. Figure 6 shows the experienced bandwidth for both flows. All these diagrams have a somewhat broken time axis in order to show sensible figures. T2 and T3 are very close together (short term fairness) and T3 and T4 are much further apart (long term fairness).

As is indicated above, each time a packet is accepted each involved counter is increased in accordance with the table above. It is not necessary that the counters are limited, but it may be practical to set an upper limit to all counters, in order to keep the size of a counter to a suitable value. In order to reflect the relationship between all counters at all times and prevent overflow, all counters must be decreased when one of the counters in a category is close to the upper limit. Thus, when a counter in a group (e.g. all TC counters under a port) reaches a limit close to the physical size, a constant is subtracted from all the counters in this group.

The operation is also cyclical with respect to time. Each time unit the variables are updated with a corresponding parameter. That is, the parameters are subtracted from the respective variable to indicate that a certain amount of time has passed and that a certain amount of traffic is sent out.

Running through all algorithms results in that flags are set. So far, no decisions have been made whether to accept or reject the packet and now it is time to use all the flags. An example of the decision sequence is listed below. When the decision is taken the respective counters are incremented and the algorithms are repeated for the next packet.

1) If port is switched off, then reject. Otherwise:
2) If Flow Groups are enabled and Flow Group is fair, then accept. Otherwise:
3) If queue (VQLP, VQLTC) longer than DiscardWanted (= desired maximum length), then reject. Otherwise:
4) If Flow Groups are enabled and queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), and the most aggressive Flow Group, then reject. Otherwise:
5) If Traffic Classes are enabled and Traffic Class is fair, then accept. Otherwise:
6) If queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), then reject. Otherwise:
7) Accept.

Below is an example of the result of the bandwidth distribution among a set of traffic classes achieved by means of the present invention. Bandwidth is measured in percent for convenience.

	Traffic class configuration			Incoming traffic	Accepted traffic		
	Guaranteed bandwidth	Weight	Maximum bandwidth		Accepted guaranteed traffic	Accepted weighted traffic	Total
Class A	10%	30	100%	80%	10%	10%	20%
Class B	20%	60	100%	10%	10%	0%	10%
Class C	5%	20	15%	40%	5%	10%	15%
Class D	5%	20	50%	10%	5%	5%	10%
Class E	0%	60	100%	10%	0%	5%	5%
Class F	0%	12	100%	50%	0%	25%	25%
Class G	0%	60	100%	50%	0%	5%	5%
Class H	0%	15	10%	100%	0%	10%	10%
Σ	40%	-	-	-	30%	70%	100%

The classes above illustrate:

If a class has less offered than guaranteed bandwidths, all get through (class B).

If a class offers more than its maximum bandwidth it is not accepted (class H).

Two classes with exactly the same input traffic, receive bandwidth according to their weights, if there is competition (classes F and G). The bandwidth is distributed in inverse proportion to the weight value in the table.

The general bandwidth calculation for a class with both a minimum and maximum bandwidth as well as a weight is:

$$B = \min(\text{offered bandwidth}, BWTC_{max}, BWTC_{min} + WTC / \Sigma WTC \times BW_{spare})$$

(The distribution between flow groups is not shown in the table.)

The method and apparatus according to the invention may be extended to more levels. Any number of levels may be added. An example is shown in figure 7. A physical connection is often shared by several users. The connection may be an overseas cable or link with a number of ports. One port may e.g. be leased to one or several telecommunication operators at Level 1of which one is shown. Only one instance is shown at each level. In turn, this operator may have a number of customers, each to be provided with a specific share of the traffic. These are located at Level 2. The customer may in turn want to control the traffic by prioritising certain users or subscribers or the like. This is done at Level 3. Below Level 3, the ordinary levels of traffic class, flow group and application flows may be as before.

At each level the traffic can be controlled using the configuration parameters including maximum bandwidth, minimum bandwidth and weight as is described above. The configuration parameters are set by the organisation in control of the specific level in question and possibly the underlying levels, if required.

The added levels are inserted above the traffic class level. An example with tables listing the configuration parameters and the respective counters and variables is set forth below with one added level. It will be appreciated that the port interfaces with the added level in the same way as with the traffic class as described above. Of course, the counters and variables have to be replaced with the corresponding counters and variables of the added level.

Similarly, the traffic class interfaces upward to the added level as with the port as described previously. Here, the counters and variables have the same name as before, but the absolute values have will have to be changed to take the added level into consideration since in fact other sets of traffic classes are located in parallel at the same level. The same applies to the flow group level and flow level.

If more levels are added, the corresponding substitutions have to be made. This is however straightforward and not described in detail here.

Configuration parameters

Port	Added Level	Traffic class	Flow group
BWP_{max}	$BWAL_{max}$	$BWTC_{max}$	
	$BWAL_{min}$	$BWTC_{min}$	
	WAL	WTC	

BWP_{max}	maximum bandwidth per port
$BWAL_{max}$	maximum bandwidth per added level
$BWAL_{min}$	minimum bandwidth per added level
WAL	weight per added level
$BWTC_{max}$	maximum bandwidth per traffic class
$BWTC_{min}$	minimum bandwidth per traffic class
WTC	weight per traffic class

Port counters and variables

Name	Logic	Increment per packet sent/discarded	Update per time unit
VQLP		+packet length	-BWP_{max}
ALP_{max}	=max(all ALs of port)	—	—
BLP_{max}	=max(all BLs of port)	—	—
CHP		+packet length × give factor, if given -packet length × WAL, if taken	× decay factor (e.g. 15/16)

VQLP	virtual queue length per port
ALP_{max}	maximum added level variable per port
BLP_{max}	maximum backlogging variable per port
CHP	charity counter for all added levels under this port

Added level counters and variables

Name	Logic	Increment per packet sent	Update per time unit
AL		+packet length × WAL	-$BWAL_{min}$ × WAL
TC_{max}	=max(all TCs of AL)	—	—
BLAL	0<BLAL<256 kB	+packet length × WAL	-$BWAL_{min}$ × WAL
VQLAL		+packet length	-$BWAL_{max}$
CHAL		+packet length × give factor, if given -packet length × WTC, if taken	× decay factor (e.g. 15/16)

AL	added level counter
TC_{max}	maximum traffic class variable per added level
BLAL	backlogging counter per added level
VQLAL	virtual queue length per added level
CHAL	charity counter for all traffic classes under this added level

Traffic Class counters and variables

Name	Logic	Increment per packet sent	Update per time unit
TC		+packet length × WTC	$-BWTC_{min} \times WTC$
FG_{max}	=max(all FGs of TC)	—	—
BLTC	0<BLTC<256 kB	+packet length × WTC	$-BWTC_{min} \times WTC$
VQLTC		+packet length	$-BWTC_{max}$

TC	traffic class counter
FG_{max}	maximum flow group variable per traffic class
BLTC	backlogging counter per traffic class
VQLTC	virtual queue length per traffic class

Flow Group counter

Name	Logic	Increment per packet sent	Update per time unit
FG		+packet length	—

FG	flow group counter

The embodiments discussed above are only intended to be illustrative of the invention. The physical implementation in hardware and software and other embodiments may be devised by those skilled in the art without departing from the spirit and scope of the following claims.

CLAIMS

1. A method for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located before output queues, the method comprising the steps of:

receiving a stream of data from the switching fabric;

subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is forwarded or interrupted (accepted or rejected).

2. A method for bandwidth scheduling according to claim 1, wherein the stream of data includes identifiable data packets;

subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected.

3. A method for bandwidth scheduling according to claim 2, wherein each packet contains information about its flow identity, namely port (number) and traffic class.

4. A method for bandwidth scheduling according to claim 3, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port.

5. A method for bandwidth scheduling according to claim 4, wherein a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

6. A method for bandwidth scheduling according to claim 5, wherein, if the counter VQLP < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

7. A method for bandwidth scheduling according to claim 4, wherein a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class.

8. A method for bandwidth scheduling according to claim 7, wherein a virtual queue is associated with each traffic class by means of a counter VQLTC (virtual queue length per traffic class) and the counter VQLTC is increased with the packet length for each accepted packet and updated each time unit by subtracting the configuration parameter $BWTC_{max}$.

9. A method for bandwidth scheduling according to claim 8, wherein, if the counter VQLTC < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

10. A method for bandwidth scheduling according to claim 3, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet, and a variable TCP_{max} (traffic class port maximum) is set to a value equal to the maximum of the TC counters for the port in question, wherein,

for a traffic class where a relationship between TC and TCP_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

11. A method for bandwidth scheduling according to claim 10, wherein the criterion between TC and TCP_{max} is the difference $TCP_{max} - TC <$ a constant.

12. A method for bandwidth scheduling according to claim 10, wherein the criterion between TC and TCP_{max} is the ratio $TC/TCP_{max} <$ a constant.

13. A method for bandwidth scheduling according to claim 3, wherein each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$).

14. A method for bandwidth scheduling according to claim 13, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet and is updated each time unit by subtracting the configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum).

15. A method for bandwidth scheduling according to claim 10, wherein a weight WTC (weight traffic class) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes.

16. A method for bandwidth scheduling according to claim 15, wherein the counter TC (traffic class) is increased with the packet length multiplied by the configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and the counter TC is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum) multiplied by the configuration parameter WTC.

17. A method for bandwidth scheduling according to claim 3, wherein, for each traffic class, a counter BL (backlogging) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class.

18. A method for bandwidth scheduling according to claim 17, wherein a variable BLP_{max} (backlogging port max) stores the maximum of the backlogging counters for the traffic classes of each port, and, for a traffic class where a relationship between BL and BLP_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

19. A method for bandwidth scheduling according to claim 18, wherein the criterion between BL and BLP_{max} is the difference $BLP_{max} - BL <$ a constant.

20. A method for bandwidth scheduling according to claim 18, wherein the criterion between BL and BLP_{max} is the ratio $BL/BLP_{max} <$ a constant.

21. A method for bandwidth scheduling according to claim 18, wherein the BL counter is increased with the packet length multiplied by a configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and

said counter BL is limited to a fixed upper value and a fixed lower value, and if one backlogging counter for a traffic class reaches the upper limit, all other counters are decreased in order to maintain the internal difference, and if one backlogging counter for a flow reaches the lower limit, this counter remains at the lower limit and the counter BL is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (minimum bandwidth per traffic class) multiplied by the configuration parameter WTC.

22. A method for bandwidth scheduling according to claim 3, wherein, if one traffic class is particularly aggressive or active, it is forced to give up a part of its accepted bandwidth.

23. A method for bandwidth scheduling according to claim 22, wherein, when a packet is accepted in a traffic class having the maximum accepted bandwidth ($TC = TCP_{max}$), a charity function forces the packet to be discarded and a counter charity (CH) is increased with a configurable fraction of the accepted packet length (+packet length × give factor), and when a packet is rejected in one of the other traffic classes, the charity function forces the packet to be accepted and the charity counter (CH) is decreased with the packet length multiplied with the weight of the respective traffic class (-packet length × WTC), and the counter (CH) is updated each time unit by multiplication with a decay factor.

24. A method for bandwidth scheduling according to claim 3, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

the bandwidth is distributed in accordance with the Max-Min algorithm;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

25. A method for bandwidth scheduling according to claim 24, wherein an added level is inserted between port level and traffic class level, said added level interfacing toward the port level as a traffic class, and interfacing toward the traffic

class level as a port.

26. A method for bandwidth scheduling according to claim 3, wherein flows are grouped together by means of a hash function into a set of flow groups.

27. A method for bandwidth scheduling according to claim 26, wherein a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group where a relationship between FG and FG_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

28. A method for bandwidth scheduling according to claim 27, wherein the criterion between FG and FG_{max} is the difference FG_{max} - FG < a constant.

29. A method for bandwidth scheduling according to claim 27, wherein the criterion between FG and FG_{max} is the ratio FG/FG_{max} < a constant.

30. A method for bandwidth scheduling according to claim 27, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

31. A method for bandwidth scheduling according to claim 30, wherein an added level is inserted between port level and traffic class level, said added level interfacing toward the port level as a traffic class, and interfacing toward the traffic class level as a port.

32. A method for bandwidth scheduling according to claim 6, 9, 10, 14, 16, 21, or 23, wherein flows are grouped together by means of a hash function into a set of flow groups, and a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group

maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group where a relationship between FG and FG_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

33. A method for bandwidth scheduling according to claim 31, wherein the criterion between FG and FG_{max} is the difference FG_{max} - FG < a constant.

34. A method for bandwidth scheduling according to claim 31, wherein the criterion between FG and FG_{max} is the ratio FG/FG_{max} < a constant.

35. A method for bandwidth scheduling according to claim 30, wherein the flags set by the algorithm logic are used in a decision sequence comprising:

if port is switched off, then reject, otherwise;

if Flow Groups are enabled and Flow Group is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardWanted (= desired maximum length), then reject, otherwise

if Flow Groups are enabled and queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), and the most aggressive Flow Group, then reject, otherwise

if Traffic Classes are enabled and Traffic Class is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), then reject, otherwise;

accept.

36. A method for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located after output queues, the method comprising the steps of:

receiving a stream of data including identifiable data packets from the switching fabric;

subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, and a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

37. A method for bandwidth scheduling according to claim 36, wherein:

a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

38. A method for bandwidth scheduling according to claim 37, wherein an added level is inserted between port level and traffic class level, said added level interfacing toward the port level as a traffic class, and interfacing toward the traffic class level as a port.

39. An arrangement for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located before output queues, the arrangement comprising:

means for receiving a stream of data from the switching fabric;

means for subjecting the stream to a decision making algorithm in the bandwidth scheduler resulting in that the stream is forwarded or interrupted (accepted or rejected).

40. An arrangement for bandwidth scheduling according to claim 39, wherein the stream of data includes identifiable data packets; and further comprising

means for subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected.

41. An arrangement for bandwidth scheduling according to claim 40, wherein each packet contains information about its flow identity, namely port (number) and traffic class.

42. An arrangement for bandwidth scheduling according to claim 41, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port.

43. An arrangement for bandwidth scheduling according to claim 42, wherein a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

44. An arrangement for bandwidth scheduling according to claim 43, wherein, if the counter VQLP < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

45. An arrangement for bandwidth scheduling according to claim 42, wherein a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class.

46. An arrangement for bandwidth scheduling according to claim 45, wherein a virtual queue is associated with each traffic class by means of a counter VQLTC (virtual queue length per traffic class) and the counter VQLTC is increased with the packet length for each accepted packet and updated each time unit by subtracting the configuration parameter $BWTC_{max}$.

47. An arrangement for bandwidth scheduling according to claim 46, wherein, if the counter VQLTC < a constant, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

48. An arrangement for bandwidth scheduling according to claim 41, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet, and a variable TCP_{max} (traffic class port maximum) is set to a value equal to the maximum of the TC counters for the port in question, and wherein, for a traffic class where a relationship between TC and TCP_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

49. An arrangement for bandwidth scheduling according to claim 48, wherein the criterion between TC and TCP_{max} is the difference $TCP_{max} - TC <$ a constant.

50. An arrangement for bandwidth scheduling according to claim 48, wherein the criterion between TC and TCP_{max} is the ratio $TC/TCP_{max} <$ a constant.

51. An arrangement for bandwidth scheduling according to claim 41, wherein each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$).

52. An arrangement for bandwidth scheduling according to claim 51, wherein a counter TC (traffic class) is increased with the packet length when the traffic class accepts a packet and is updated each time unit by subtracting the configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum).

53. An arrangement for bandwidth scheduling according to claim 48, wherein a weight WTC (weight traffic class) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes.

54. An arrangement for bandwidth scheduling according to claim 53, wherein the counter TC (traffic class) is increased with the packet length multiplied by the configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and the counter TC is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (bandwidth per traffic class minimum) multiplied by the configuration parameter WTC.

55. An arrangement for bandwidth scheduling according to claim 41, wherein, for each traffic class, a counter BL (backlogging) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more

bandwidth to this traffic class.

56. An arrangement for bandwidth scheduling according to claim 55, wherein a variable BLP_{max} (backlogging port max) stores the maximum of the backlogging counters for the traffic classes of each port, and, for a traffic class where a relationship between BL and BLP_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet.

57. An arrangement for bandwidth scheduling according to claim 56, wherein the criterion between BL and BLP_{max} is the difference $BLP_{max} - BL <$ a constant.

58. An arrangement for bandwidth scheduling according to claim 56, wherein the criterion between BL and BLP_{max} is the ratio $BL/BLP_{max} <$ a constant.

59. An arrangement for bandwidth scheduling according to claim 55, wherein the counter BL is increased with the packet length multiplied by a configuration parameter WTC (weight traffic class), when the traffic class accepts a packet, and said counter BL is limited to a fixed upper value and a fixed lower value, and if one backlogging counter for a traffic class reaches the upper limit, all other counters are decreased in order to maintain the internal difference, and if one backlogging counter for a flow reaches the lower limit, this counter will remain at the lower limit and the counter BL is updated each time unit by subtracting a configuration parameter $BWTC_{min}$ (minimum bandwidth per traffic class) multiplied by the configuration parameter WTC.

60. An arrangement for bandwidth scheduling according to claim 41, wherein, if one traffic class is particularly aggressive or active, it is forced to give up a part of its accepted bandwidth.

61. An arrangement for bandwidth scheduling according to claim 60, wherein, when a packet is accepted in a traffic class having the maximum accepted bandwidth ($TC = TCP_{max}$), a charity function forces the packet to be discarded and a counter charity (CH) is increased with a configurable fraction of the accepted packet length (+packet length × give factor), and when a packet is rejected in one of the other traffic classes, the charity function forces the packet to be accepted and the charity counter (CH) is decreased with the packet length multiplied with the weight of the respective traffic class (-packet length × WTC), and the counter (CH) is updated each time unit by multiplication with a decay factor.

62. An arrangement for bandwidth scheduling according to claim 41, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

the bandwidth is distributed in accordance with the Max-Min algorithm;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

63. An arrangement for bandwidth scheduling according to claim 62, wherein an added level is inserted between port level and traffic class level, said added level interfacing toward the port level as a traffic class, and interfacing toward the traffic class level as a port.

64. An arrangement for bandwidth scheduling according to claim 41, wherein flows are grouped together by means of a hash function into a set of flow groups.

65. An arrangement for bandwidth scheduling according to claim 64, wherein a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group where a relationship between FG and FG_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

66. An arrangement for bandwidth scheduling according to claim 65, wherein the criterion between FG and FG_{max} is the difference $FG_{max} - FG <$ a constant.

67. An arrangement for bandwidth scheduling according to claim 65, wherein the criterion between FG and FG_{max} is the ratio $FG/FG_{max} <$ a constant.

68. An arrangement for bandwidth scheduling according to claim 65, wherein:

a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, a virtual queue is associated with each port, and a flag is set in dependence of the queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes:

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

69. An arrangement for bandwidth scheduling according to claim 68, wherein an added level is inserted between port level and traffic class level, said added level interfacing toward the port level as a traffic class, and interfacing toward the traffic class level as a port.

70. An arrangement for bandwidth scheduling according to claim 44, 47, 50, 52, 54, 59, or 61, wherein flows are grouped together by means of a hash function into a set of flow groups, and a counter FG (flow group) is increased with the packet length when the flow group accepts a packet, and a variable FG_{max} (flow group maximum) is set to a value equal to the maximum of the FG counters for the traffic class in question, and wherein, for a flow group where a relationship between FG and FG_{max} meets a criterion, a flag is set to a value used by the algorithm in deciding to accept or reject a packet, whereby the bandwidth is distributed in accordance with the Max-Min algorithm.

71. An arrangement for bandwidth scheduling according to claim 70, wherein the criterion between FG and FG_{max} is the difference $FG_{max} - FG <$ a constant.

72. An arrangement for bandwidth scheduling according to claim 70, wherein the criterion between FG and FG_{max} is the ratio $FG/FG_{max} <$ a constant.

73. An arrangement for bandwidth scheduling according to claim 68, wherein the flags set by the algorithm logic are used in a decision sequence comprising:

if port is switched off, then reject, otherwise;

if Flow Groups are enabled and Flow Group is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardWanted (= desired maximum length), then reject, otherwise

if Flow Groups are enabled and queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), and the most aggressive Flow Group, then reject, otherwise

if Traffic Classes are enabled and Traffic Class is fair, then accept, otherwise;

if queue (VQLP, VQLTC) longer than DiscardPreferred (= preferred maximum length), then reject, otherwise;

accept.

74. An arrangement for bandwidth scheduling in a switch comprising a switching fabric, and a bandwidth scheduler located after output queues, the

arrangement comprising:

means for receiving a stream of data including identifiable data packets from the switching fabric;

means for subjecting each data packet to a decision making algorithm in the bandwidth scheduler resulting in that the data packet is accepted or rejected, wherein a limit (BWP_{max}) is set on the maximum accepted bandwidth per port, and a virtual queue is associated with each port by means of a counter VQLP (virtual queue length of port) and the counter VQLP is increased with the packet length for each accepted packet and updated by subtracting the configuration parameter BWP_{max} (maximum accepted bandwidth per port) each time unit.

75. An arrangement for bandwidth scheduling according to claim 74, wherein:

a flag is set in dependence of the port queue length;

a limit ($BWTC_{max}$) is set on the maximum accepted bandwidth per traffic class, a virtual queue is associated with each traffic class, and a flag is set in dependence of the traffic class queue length;

each traffic class is guaranteed a bandwidth up to a limit ($BWTC_{min}$);

a weight (WTC) is associated with each traffic class, so that the algorithm automatically prioritises certain traffic classes;

for each traffic class, a backlogging counter (BL) keeps track of how many packets are accepted in relation to the other traffic classes, so that if a previously idle traffic class becomes active, the traffic class is compensated by distributing more bandwidth to this traffic class;

if one traffic class is particularly aggressive or active, it gives up a part of its accepted bandwidth.

76. An arrangement for bandwidth scheduling according to claim 75, wherein an added level is inserted between port level and traffic class level, said added level interfacing toward the port level as a traffic class, and interfacing toward the traffic class level as a port.

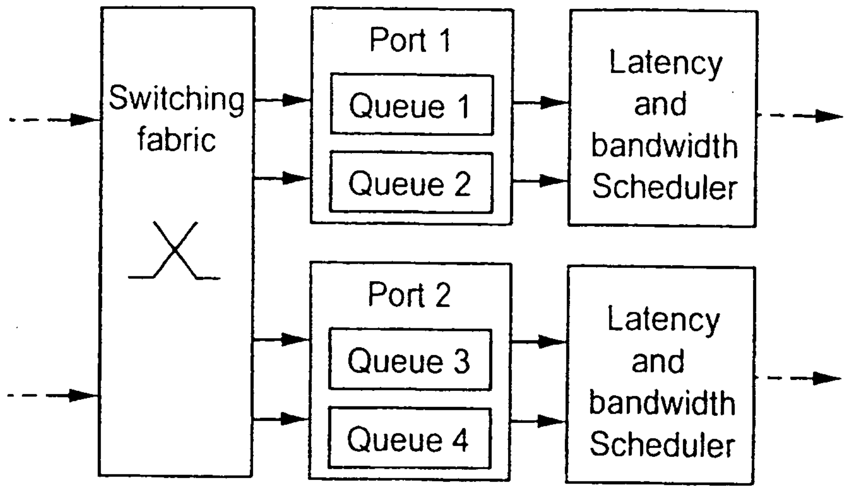

Switching fabric
Port 1
Queue 1
Queue 2
Latency and bandwidth Scheduler
Port 2
Queue 3
Queue 4
Latency and bandwidth Scheduler

Fig.1 prior art

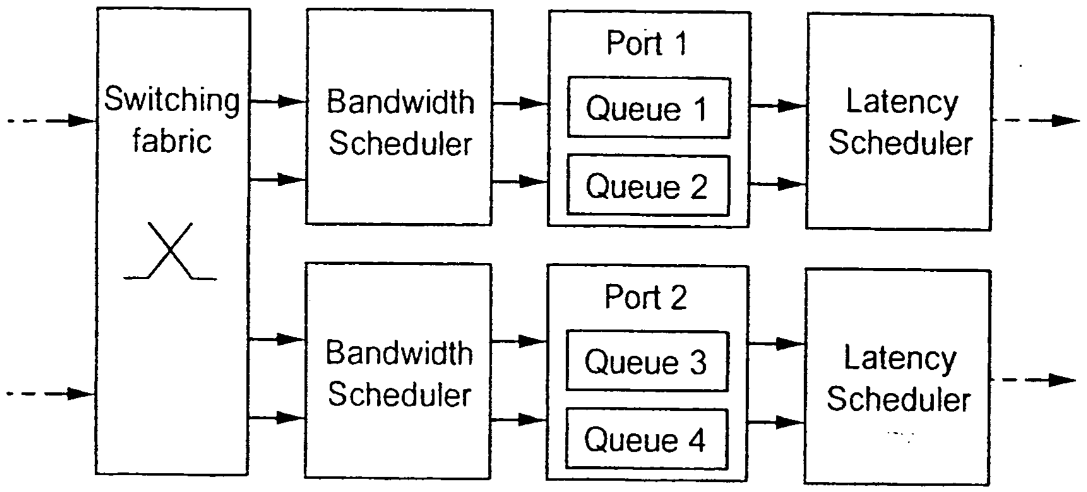


Fig.2

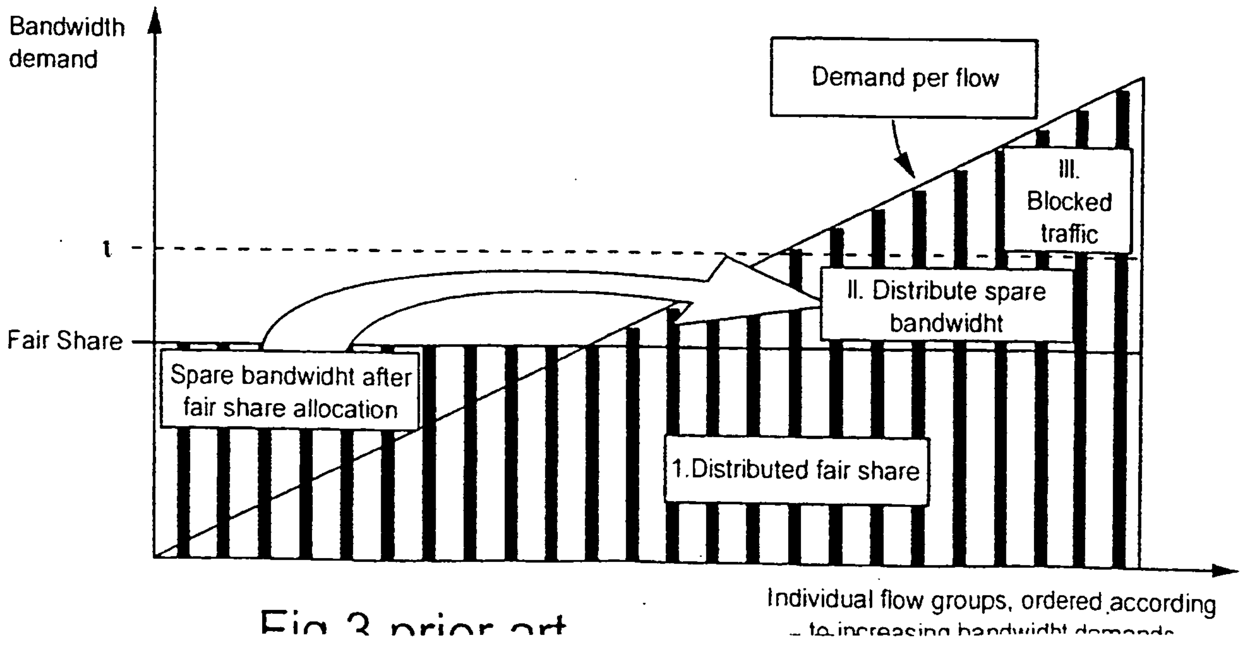


Fig.3 prior art

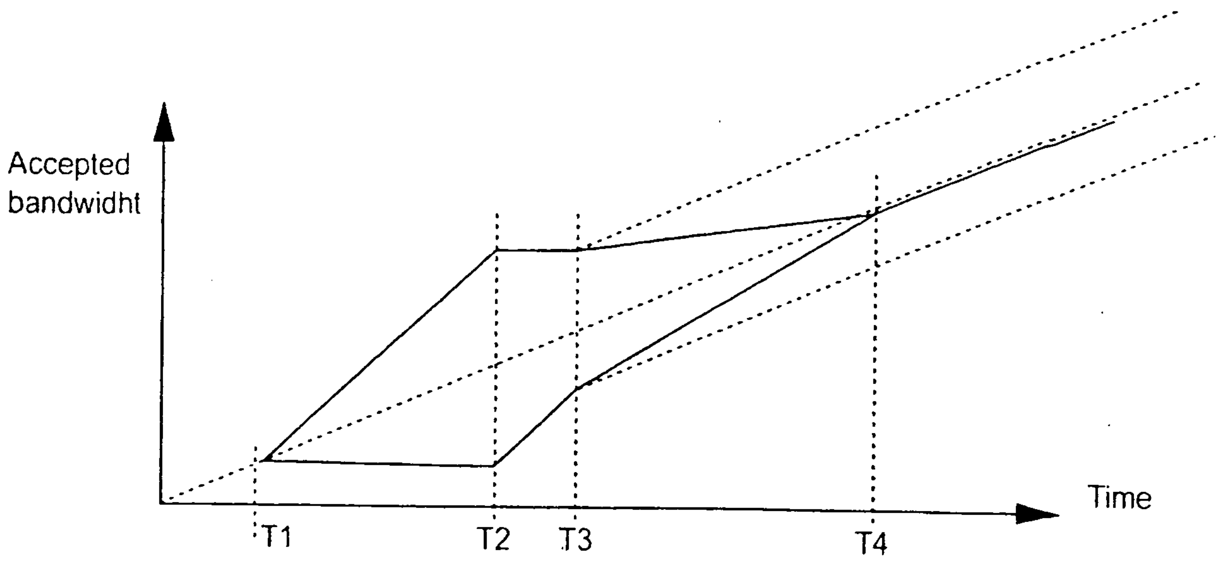


Fig.4

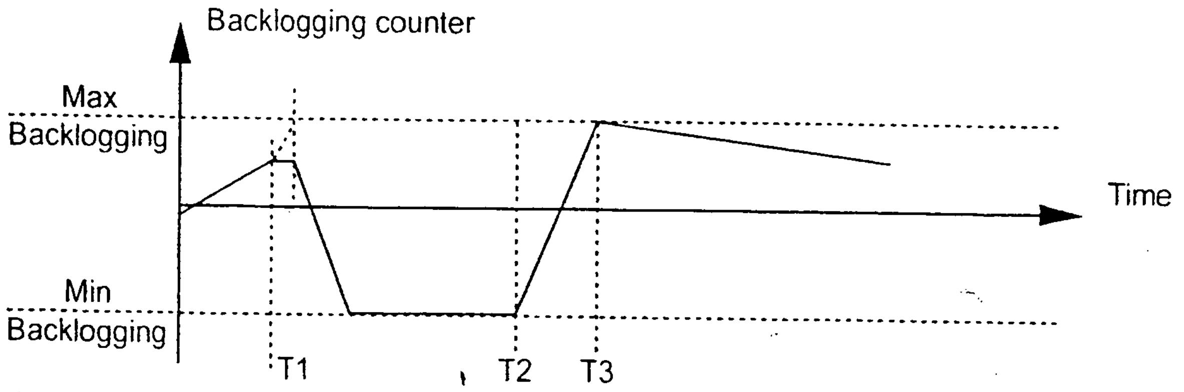


Fig.5

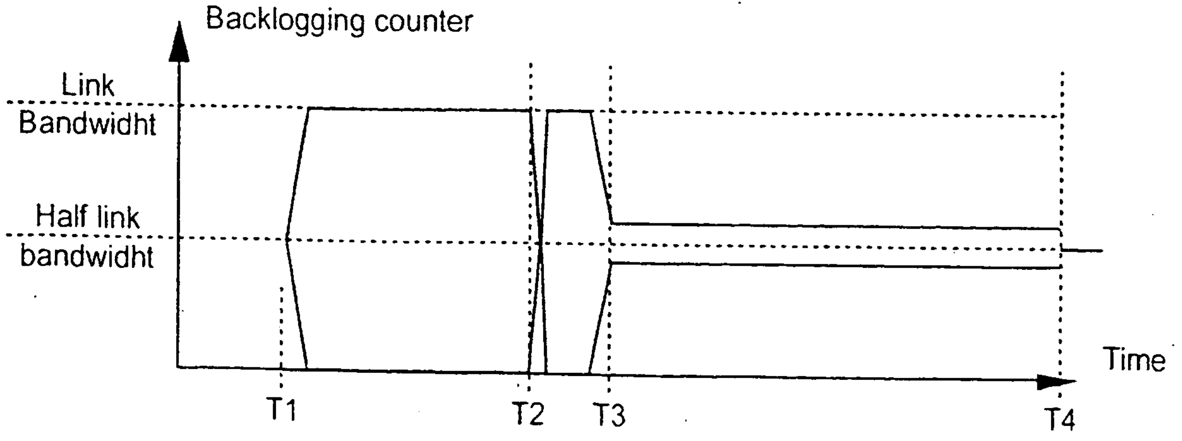


Fig.6

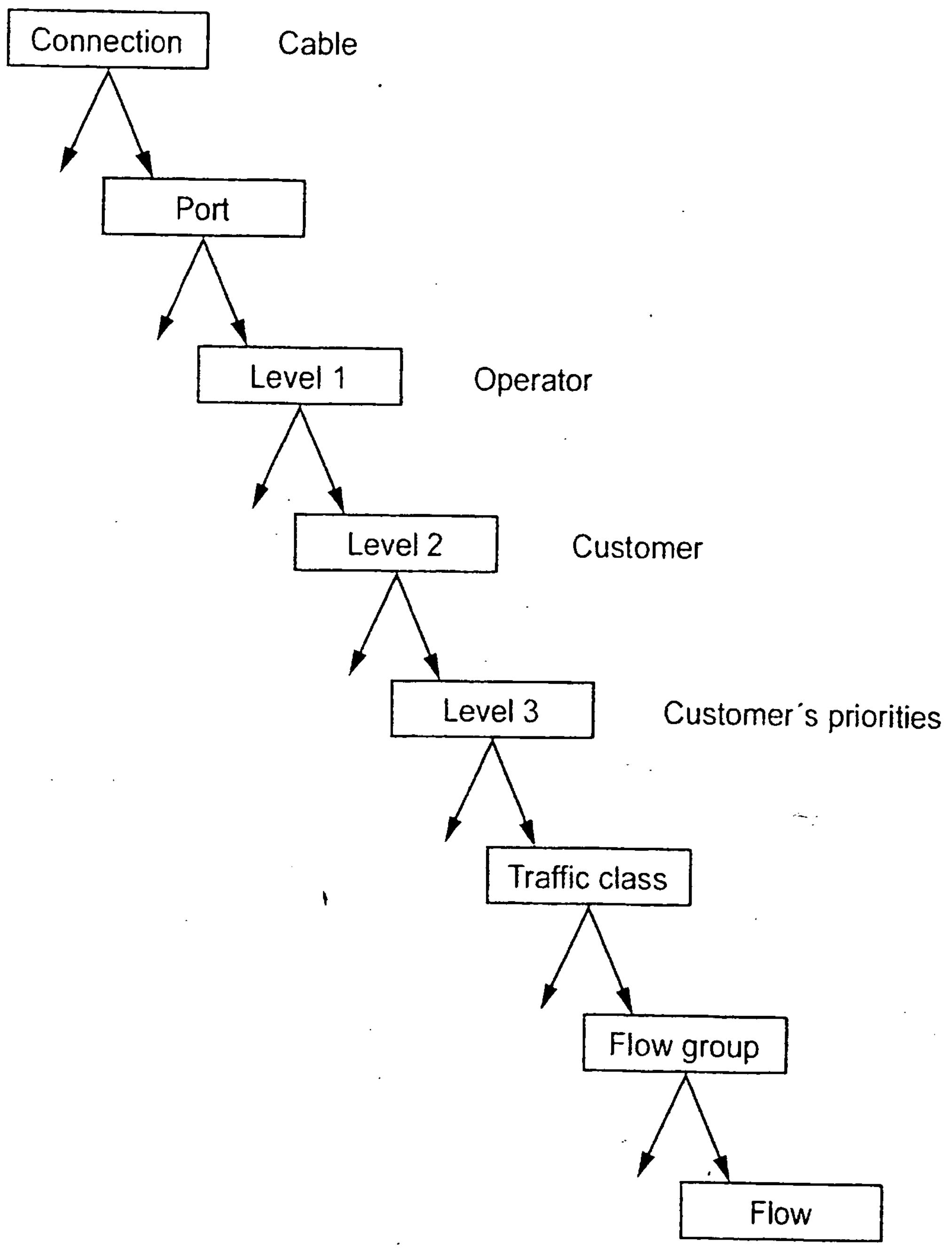

Connection
Cable
Port
Level 1
Operator
Level 2
Customer
Level 3
Customer´s priorities
Traffic class
Flow group
Flow

Fig.7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 01/00733

A. CLASSIFICATION OF SUBJECT MATTER

IPC7: H04Q 3/64, H04L 12/56
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7: H04L, H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-INTERNAL, WPI DATA, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	DEBBIE ROBINSON "Wire Speed Quality of Service over Ethernet" 2000-05-19 [online] [retrieved on 2001-07-11] Retrieved from the Internet: <URL: http://www.chipcenter.com/networking/techarch.html> chapter 3 --	1-76
A	GOYAL, P et al "Fair airport scheduling algorithms" In: Proceedings of the IEEE 7th International Workshop on Network and Operating System Support for Digital Audio and Video, 1997, 19-21 May 1997, p. 257-265; chapter 1; chapter 5 - chapter 6; abstract --	1-76

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 17 July 2001	Date of mailing of the international search report 20-07-2001
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM	Authorized officer RICKARD ELG/EE

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 01/00733

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IEICE TRANS. COMMUN., Volume E81-B, No 5, May 1998, YIWEI THOMAS HOU ET AL, "ATM ABR Traffic Control with a Generic Weight-Based Bandwidth Sharing Policy: Theory and a Simple Implementation", summary, chapter 1 - chapter 3 --	1-76
A	YUAN WU et al"Improved virtual queueing and dynamic EPD techniques for TCP over ATM" In: Proceedings of International Conference on Network Protocols, 1997.28-31 Oct. 1997, p. 212-219 page 214 - page 215; chapter III --	1-76
A	GUIJARRO, L. et al "Guaranteeing fairness and protection in ABR by means of fair queueing" In: 2nd International Conference on ATM, 1999. ICATM '99. 21-23 June 1999, p. 22-31; page 29- page 30; chapter 5 --	1-76
A	RACZ, A. et al "Weighted fair early packet discard at an ATM switch output port" In: Proceedings. IEEE Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM '99 21-25 March 1999, p. 1160-1168 vol. 3 chapter V, abstract --	1-76
A	HUI ZHANG "Service disciplines for guaranteed performance service in packet-switching networks" In: Proceedings of the IEEE, Oct. 1995, p. 1374-1396; page 1392; chapter I --	1-76
A	US 5764641 A (ARTHUR LIN), 9 June 1998 (09.06.98), claims 1-6 --	1-76
A	US 5768257 A (TODD L. KHACHERIAN ET AL), 16 June 1998 (16.06.98), figure 3, abstract --	1-76

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SE 01/00733

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5748629 A (STEPHEN A. CALDARA ET AL), 5 May 1998 (05.05.98), column 8, line 41 - column 10, line 45, figures 6-7, claims 13,23-28	1-76

--

INTERNATIONAL SEARCH REPORT
Information on patent family members

02/07/01

International application No.
PCT/SE 01/00733

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5748629 A	05/05/98	AU 6500796 A	18/02/97
		AU 6500896 A	18/02/97
		AU 6500996 A	18/02/97
		AU 6501096 A	18/02/97
		AU 6501496 A	18/02/97
		AU 6501696 A	18/02/97
		AU 6501796 A	18/02/97
		AU 6501996 A	18/02/97
		AU 6502096 A	18/02/97
		AU 6502496 A	18/02/97
		AU 6502596 A	18/02/97
		AU 6502696 A	18/02/97
		AU 6502796 A	18/02/97
		AU 6503196 A	18/02/97
		AU 6503296 A	18/02/97
		AU 6503396 A	18/02/97
		AU 6503496 A	18/02/97
		AU 6503596 A	18/02/97
		AU 6503696 A	18/02/97
		AU 6503796 A	18/02/97
		AU 6549196 A	18/02/97
		AU 6549296 A	18/02/97
		AU 6648496 A	18/02/97
		AU 6648796 A	18/02/97
		AU 6712496 A	18/02/97
		AU 6712596 A	18/02/97
		AU 6761896 A	18/02/97
		AU 6762096 A	18/02/97
		EP 0839419 A	06/05/98
		EP 0839420 A	06/05/98
		EP 0839421 A	06/05/98
		EP 0839422 A	06/05/98
		EP 0845181 A	03/06/98
		EP 0872086 A	21/10/98
		JP 11510003 T	31/08/99
		JP 11510004 T	31/08/99
		JP 11510005 T	31/08/99
		JP 11510006 T	31/08/99
		JP 11510007 T	31/08/99
		JP 11510008 T	31/08/99
		JP 11510009 T	31/08/99
		JP 11510010 T	31/08/99
		JP 11510011 T	31/08/99
		JP 11510012 T	31/08/99
		JP 11510013 T	31/08/99
		JP 11510014 T	31/08/99
		JP 11510323 T	07/09/99
		JP 11510324 T	07/09/99
		JP 11510327 T	07/09/99
		JP 11510328 T	07/09/99
		JP 11510329 T	07/09/99
		JP 11510330 T	07/09/99
		JP 11510331 T	07/09/99
		JP 11511303 T	28/09/99
		JP 2000501897 T	15/02/00

INTERNATIONAL SEARCH REPORT
Information on patent family members

02/07/01

International application No.
PCT/SE 01/00733

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5748629 A	05/05/98	JP 2000501900 T	15/02/00
		JP 2000501901 T	15/02/00
		JP 2000501902 T	15/02/00
		JP 2001500323 T	09/01/01
		US 5781533 A	14/07/98
		US 5787086 A	28/07/98
		US 5790770 A	04/08/98
		US 5822540 A	13/10/98
		US 5850395 A	15/12/98
		US 5862137 A	19/01/99
		US 5867663 A	02/02/99
		US 5870538 A	09/02/99
		US 5872769 A	16/02/99
		US 5889956 A	30/03/99
		US 5896511 A	20/04/99
		US 5905729 A	18/05/99
		US 5909427 A	01/06/99
		US 5917805 A	29/06/99
		US 5933429 A	03/08/99
		US 5948067 A	07/09/99
		US 5956342 A	21/09/99
		US 5978359 A	02/11/99
		US 5982771 A	09/11/99
		US 5982776 A	09/11/99
		US 5983260 A	09/11/99
		US 5996019 A	30/11/99
		US 6002667 A	14/12/99
		US 6076112 A	13/06/00
		US 6088736 A	11/07/00
		US 6115748 A	05/09/00
		US 6141346 A	31/10/00
		US 6167452 A	26/12/00
		US 6236655 B	22/05/01
		WO 9703549 A	06/02/97
		WO 9704397 A	06/02/97
		WO 9704541 A	06/02/97
		WO 9704542 A	06/02/97
		WO 9704543 A	06/02/97
		WO 9704544 A	06/02/97
		WO 9704546 A	06/02/97
		WO 9704548 A	06/02/97
		WO 9704549 A	06/02/97
		WO 9704552 A	06/02/97
		WO 9704554 A	06/02/97
		WO 9704555 A	06/02/97
		WO 9704556 A	06/02/97
		WO 9704557 A	06/02/97
		WO 9704558 A	06/02/97
		WO 9704559 A	06/02/97
		WO 9704560 A	06/02/97
		WO 9704561 A	06/02/97
		WO 9704562 A	06/02/97
		WO 9704563 A	06/02/97
		WO 9704564 A	06/02/97
		WO 9704565 A	06/02/97

INTERNATIONAL SEARCH REPORT
Information on patent family members

02/07/01

International application No.
PCT/SE 01/00733

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5748629 A	05/05/98	WO 9704566 A	06/02/97
		WO 9704567 A	06/02/97
		WO 9704568 A	06/02/97
		WO 9704569 A	06/02/97
		WO 9704570 A	06/02/97
		WO 9704571 A	06/02/97

INTERNATIONAL SEARCH REPORT
Information on patent family members

02/07/01

International application No.
PCT/SE 01/00733

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5764641 A	09/06/98	NONE	
US 5768257 A	16/06/98	NONE	